СПИСОК ЛИТЕРАТУРЫ

1. IEEE Standard for Local and metropolitan area networks Part 16: Air Interface for Broadband Wireless Access Systems // IEEE Std 802.16–2009 (Revision of IEEE Std 802.16–2004). PP. C1–2004, May 29, 2009. doi: 10.1109/IEEESTD.2009.5062485). URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=5062485&isnumber=5062484, (дата обращения: 04.09.2012).
2. Вишневский В.М., Портной С.Л., Шахнович И.В. Энциклопедия WiMAX. Путь к 4G. – М.: Техносфера, 2009. – 472 с.
3. Taylor H., Yochem A., Phillips L., Martinez F. Event-Driven Architecture: How SOA Enables the Real-Time Enterprise. – Boston: Addison-Wesley, 2009. – 308 p.
4. UniTESK. URL: http://unitesk.ru/ (дата обращения: 04.09.2012).
5. Грошев С.Г. Применение технологии UniTesK для тестирования систем с различной конфигурацией активных потоков управления. // citforum.ru. 2012. URL: http://citforum.ru/SE/testing/unitest_use/ (дата обращения: 04.09.2012).
6. Nuaymi L. WiMAX: Technology for Broadband Wireless Access. – New York: John Wiley & Sons, 2007. – 310 p.

Поступила 14.09.2012 г.

УДК 004.021

ЭКСПЕРИМЕНТ ПО ФРАКТАЛЬНОМУ СЖАТИЮ RGB-ИЗОБРАЖЕНИЙ НА ВЫЧИСЛИТЕЛЬНОМ КЛАСТЕРЕ

И.В. Бойченко, С.С. Кулбаев, А.А. Немеров, В.В. Голенков

Томский государственный университет систем управления и радиоэлектроники
E-mail: kulbaev@asu.tusur.ru

Описан эксперимент по сжатию полноцветных изображений на основе фракталов с применением высокопроизводительной вычислительной системы с распределенной памятью – вычислительном кластере. Межпроцессный обмен осуществляется на основе технологии MPI. Показана линейная зависимость времени вычислений от количества вычислительных процессов. Выявлена неравномерность нагрузки вычислительных процессов, вызванная неоднородностью сжимаемых изображений. Проведено сравнение качества и размера сжимаемых изображений на основе фракталов и на основе алгоритма JPEG.

Ключевые слова:
Параллельные вычисления, интерфейс передачи сообщений, фракталы, сжатие изображений, вычислительные процессы, высокопроизводительные вычислительные системы.

Key words:
Parallel computation, message passing interface, fractals, image compression, computational processes, high-performance computing systems.

Повышенный интерес к алгоритмам сжатия изображений на основе фракталов вызван необходимостью минимизации размеров передаваемых или хранимых данных. Для сжатия статических изображений существует множество методов [1–3]. Наиболее известными из методов сжатия графической информации с потерей качества являются алгоритмы JPEG и JPEG2000. Поэтому, при разработке новых алгоритмов, проводят сравнение именно с JPEG. При рассмотрении алгоритмов сжатия наиболее важными являются три основных свойства: коэффициент компрессии, степень потери качества по сравнению с оригиналом, трудоемкость. Однако в рамках различных задач ценность этих параметров не равнозначна. Для случая дефицита дискового пространства и полосы пропускания каналов, при достаточном количестве вычислительных мощностей, представляет интерес использование фрактального сжатия [2], обладающего потенциально более высоким коэффициентом компрессии, но более трудоемким при сжатии. Обратный процесс – распаковка, требует меньше вычислений, чем у JPEG [3]. Свойство, заключающееся в независимости восстанавливаемого изображения от разрешения, позволяет использовать фрактальный алгоритм для визуализации цифровых изображений на больших экранах.

Как правило, в работах, посвященных исследованию алгоритмов фрактального сжатия, накладывается такое ограничение, что алгоритм должен обеспечивать приемлемое время сжатия на персональном компьютере [4, 5]. Поскольку полный перебор при поиске соответствия доменных и ранговых блоков приводит к большому количеству операций, то усилия разработчиков, в основном, направлены на сокращение количества сравнений блоков за счет предварительной классификации [6]. Сокращение числа рассматриваемых блоков неизбежно приводит к большим потерям качества по сравнению с полным перебором. В данном исследовании ограничение на производительность вычислительной системы было ослаблено исходя из того, что высокопроизводительные системы разных классов становятся все более доступными для конечных пользователей. Так, сервисы типа GRID и «облачные вычисления» позволяют задействовать удаленные вычислительные мощности в режиме on-line. Другим трендом развития совре-

менных вычислительных систем являются акселераторы вычислений GPGPU, ярким представителем которых является технология CUDA, обеспечивающая терафлопную производительность за сравнительно низкую стоимость. Основной целью эксперимента являлась проверка масштабируемости алгоритма фрактального сжатия на системах с распределенной памятью. Системы с распределенной памятью позволяют объединять большое количество вычислителей – десятки и сотни тысяч ядер для передовых представителей класса.

В качестве исследуемого алгоритма был выбран базовый алгоритм фрактального сжатия изображений. Для поиска доменных блоков применялся метод полного перебора без предварительной классификации.

Базовый алгоритм фрактального сжатия

Понятие фрактал означает, что целое изображение состоит из уменьшенных копий его самого или его частей, то есть обладает свойством самоподобия. Увеличивая такое изображение, мы можем наблюдать одну и ту же степень детализации независимо от разрешения.

Алгоритм фрактального сжатия, предложенный Майклом Барнсли [7], основан на системе итерируемых функций (IFS –Iteration Function System). Но IFS не могут быть использованы как готовые системы сжатия реалистичных изображений. IFS необходима для понимания того, как работают фрактальные методы сжатия изображений. Арнауд Джеквин [8] впервые представил метод фрактального кодирования, основанный на IFS, в котором используются системы доменных и ранговых блоков реалистичных изображений.

Базовый алгоритм фрактального кодирования изображения описывается следующим образом:

1. Изображение $M \times N$ пикселей разбивается на множество R-блоков (ранговые блоки (r)) $R_1, R_2, ..., R_r$, где R_i, i=1,..., r есть квадратный $B \times B$ пиксельный фрагмент изображения.
2. Изображение покрывается последовательностью D-блоков (доменные блоки) $D_1, D_2, ..., D_d$, где D_i – представляющие квадратные (возможно пересекающиеся) $2B \times 2B$ пиксельные фрагменты. Домены могут быть разного размера, и их количество может исчисляться сотнями и тысячами.
3. Для каждого рангового блока находят домен и соответствующее преобразование, которое наилучшим образом покрывает ранговый блок. Обычно это аффинное преобразование.
4. Если достаточно точного соответствия не получилось, то разбиваем ранговые блоки на меньшие блоки. Данный процесс продолжается до тех пор, пока не получают приемлемого соответствия при заданной допустимой погрешности, или размер рангового блока достигает заданного минимального значения.

Организация вычислительных процессов фрактального сжатия

Одним из главных факторов увеличения времени сжатия является количество доменов. Каждый выбранный ранговый блок (с восьмью видами аффинного преобразования) необходимо сопоставить со всеми доменными блоками. Для уменьшения процесса перебора была предложена классификация доменных и ранговых блоков [9]. Так же процесс вычисления можно оптимизировать так, чтобы каждый ранговый блок обрабатывался по отдельности, то есть данные могут обрабатываться независимо, что позволяет распараллелить процесс вычисления [10].

В [6] представлен быстродействующий алгоритм фрактального сжатия изображений с глубиной цветности 8 бит. Ускорение достигается за счет уменьшения количества доменов.

В [11] представлена параллельная организация вычислительных процессов, основанная на многопоточной обработке, то есть в системах с общей памятью.

В данной статье предлагается параллельная реализация фрактального сжатия цветных изображений (глубина цветности 24 бит) с применением технологии MPI (**Message Passing Interface –** интерфейс передачи сообщений) [12]. В реализации использована модификация алгоритма из [2, 3] с переупорядочиванием блоков пикселей в памяти и разбиением методом квадродерева. Классификация доменных и ранговых блоков не использовалась, так как сокращение множества доменных блоков приводит к разбросу значения качества выходного (восстановленного) изображения.

Масштабирование параллельных вычислений с помощью MPI

В предлагаемой параллельной реализации алгоритма фрактального сжатия изображений вычислительные процессы (количеством N) организованы следующим образом:

1) главный процесс (с номером 0):
   - загружает и передает изображение в каждый рабочий процесс;
   - создает список задач (список ранговых блоков (R)) и распределяет задачи между рабочими процессами;
   - получает результат от очередного рабочего процесса и записывает в файл.
2) каждый рабочий процесс (с номерами от 1 до N–1):
   - получает изображение и создает список доменов для сравнения с ранговыми блоками;
   - получает список задач (список ранговых блоков) для поиска соответствий с доменными блоками;
   - производит вычисления (сопоставление доменных блоков с ранговыми блоками);
   - записывает параметры найденного соответствующего доменного блока в буфер, затем буфер передает *главному процессу*;

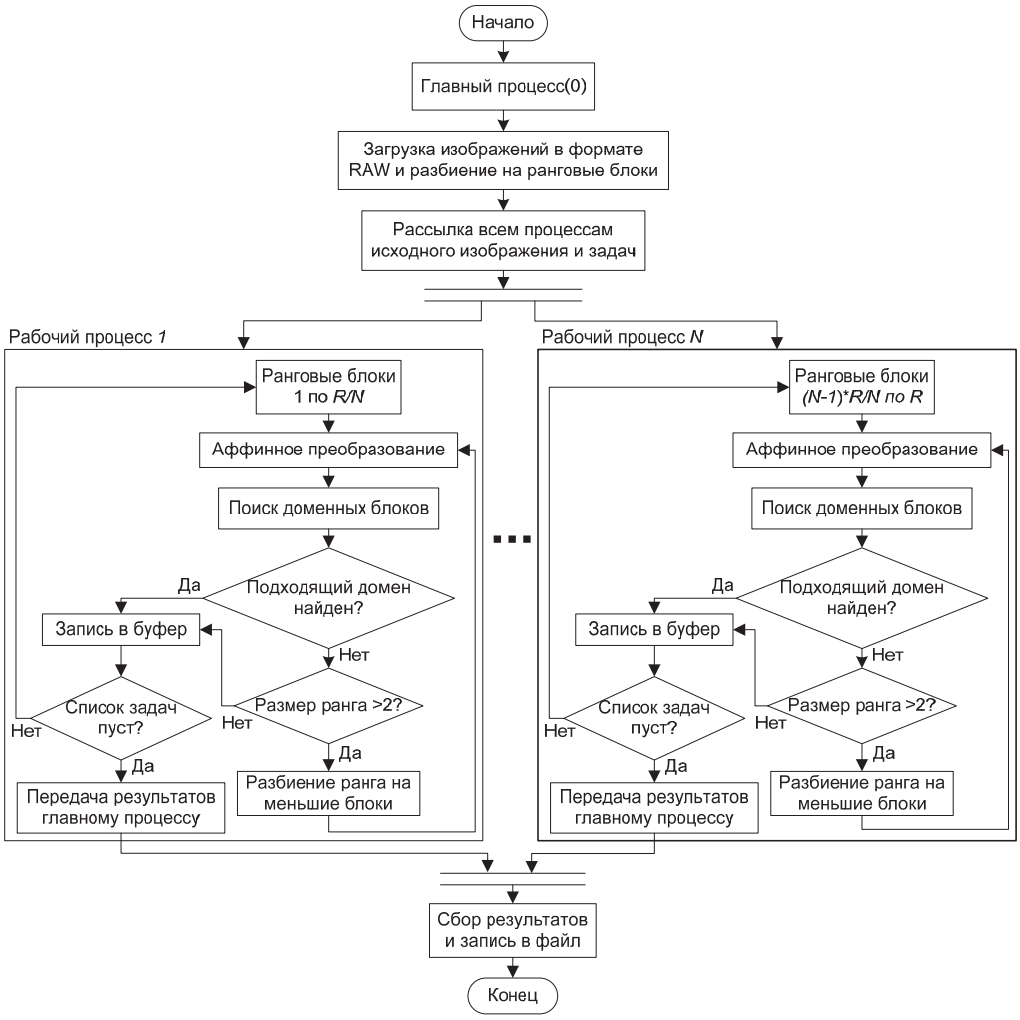

Рис. 1. *Блок-схема параллельного алгоритма сжатия изображений на основе фракталов*

- в случае отсутствия соответствующего доменного блока для выбранного ранга разбивает ранговый блок на меньшие блоки методом квадродерева и повторяет вычисления.

На рис. 1 приведена блок-схема параллельного алгоритма фрактального сжатия. Реализация алгоритма фрактального сжатия на нескольких вычислительных узлах и независимость ранговых блоков позволяют обеспечить хорошую масштабируемость задачи. В программной реализации для обмена сообщениями между главным процессом и рабочими процессами была применена технология MPI.

Создание списка доменных блоков в каждом рабочем процессе освобождает от необходимости обмена доменными блоками по сети, но повышает требуемый объем оперативной памяти. Каждое разбиение ранговых блоков на меньшие блоки назовем итерацией. Тогда, при количестве рабочих процессов, равном количеству рангов, сжатие одного блока займет от 1 до 4 итераций в случае ранга 16×16 пикселей, и от 1 до 3 итераций, в случае ранга 8×8 пикселей. Изображение разбивается на большее количество блоков на тех участках, где имеется больше деталей для кодирования. Следовательно, время вычисления будет зависеть от структуры изображения и от времени обращения к памяти.

Контроль качества восстанавливаемого изображения

Качество восстанавливаемого изображения зависит от входных задаваемых параметров алгоритма фрактального сжатия. Параметром, сильно влияющим на качество, является пороговое значение, задающее разницу между обрабатываемым фрагментом и наилучшим его приближением.

Чем меньше допустимая погрешность, тем точнее осуществляется поиск доменного блока. С другой стороны, увеличивается количество операций, что влияет на общее время выполнения вычислений. Так же на качество восстанавливаемого изображения влияет шаг поиска доменов и размер ранговых блоков. Чем меньше шаг поиска, тем больше количество доменов и больше вероятность нахождения соответствующего доменного блока. Размер ранговых блоков влияет на размер выходного файла.

Чем меньше алгоритм дробит входное изображение на ранговые блоки, тем меньше количество хранимых структур доменных блоков, что уменьшает размер выходного файла. Для оценки качества восстанавливаемого изображения были использованы метрики SSIM и PSNR [13].

Результаты вычислительного эксперимента

Реализация алгоритма тестировалась на вычислительном кластере МСЦ РАН, предоставленном по программе «Университетский кластер» [14]. Установка состоит из 64-х вычислительных узлов, каждый из которых включает в себя 2 процессора Intel®Xeon®E5450 с тактовой частотой 3 ГГц, 8 Гб оперативной памяти и 100 Гб внешней памяти. Вычислительные узлы объединены сетью Infiniband. Кластер работает под управлением ОС Linux CentOS 5.4. Выполнение задач осуществляется через систему пакетной обработки TORQUE и планировщик ресурсов MAUI.

Исследуемый алгоритм был реализован на языке Си с применением библиотеки MPI версии 2.1.

Для тестов использовались цветные (RGB) изображения из базы данных группы фрактального кодирования и анализа (fractal coding and analysis group) [15]. Используемые далее характеристики включают имя входного файла, количество используемых процессов (ядер), время сжатия, время декодирования (восстановления), размер сжатого файла, качество восстановленного по метрикам SSIM и PSNR.

В состав параметров фрактального алгоритма входят:

- размер рангового блока (начальный размер 8×8 пикселей);
- шаг поиска домена (4 пикселя);
- предел допустимой погрешности (0,005).

Предел допустимой погрешности определяет расстояние между средними значениями ранговых и доменных блоков. Если разница между средними значениями равна нулю, то сопоставляемый доменный блок идентичен ранговому блоку. Так как алгоритм не гарантирует нахождения идентичного доменного блока для всех ранговых блоков, то целесообразно задать некую допустимую погрешность при поиске. Для оценивания расстояния между блоками использована функция среднеквадратического отклонения [2, 9]:

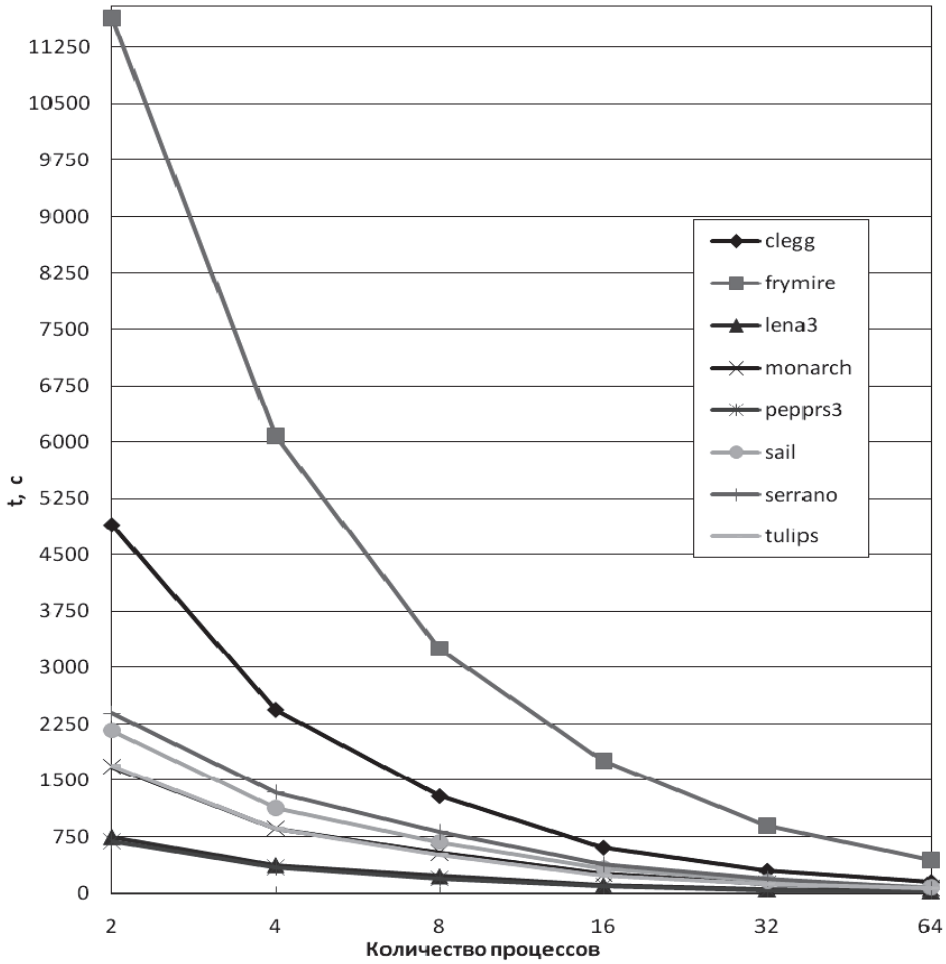

Рис. 2. *Зависимость времени работы параллельного алгоритма от количества задействованных процессоров*

Таблица. *Результаты испытания MPI-реализации параллельного алгоритма фрактального сжатия*

Изображение	Фрактальный алгоритм						Алгоритм JPEG		
	Время сжатия на 64-х ядрах, с	Время сжатия на 1 ядре, с	Время декодера, с	Размер файла, кб	SSIM	PSNR, дБ	Размер файла, кб	SSIM	PSNR, дБ
clegg	155,529	4894,304	0,240	167,800	0,998	31,326	366,800	0,994	31,110
frymire	443,582	11642,376	0,250	584,100	0,989	30,690	601,900	0,997	33,390
lena3	28,484	743,089	0,051	79,300	0,987	38,243	74,600	0,985	39,753
monarch	66,466	1683,099	0,081	124,500	0,998	40,813	103,300	0,992	44,950
peppers3	24,361	686,439	0,054	68,400	0,989	39,753	72,700	0,944	42,076
sail	79,114	2164,089	0,104	197,100	0,994	38,620	163,000	0,991	42,090
serrano	64,239	2389,876	0,103	128,600	0,989	34,288	205,700	0,923	34,823
tulips	63,304	1697,637	0,087	117,100	0,991	39,083	129,400	0,985	42,100

$$\sigma = \sum_{i=1}^{N} (R_i^* - D_i^*)^2 / N,$$

где R_i^* – i-й преобразованный ранговый блок (аффинные преобразования); B_i^* – i-й преобразованный доменный блок (D_i^*=sD_i+o, где s – коэффициент изменения контраста; o – коэффициент сдвига по яркости); N – количество пикселей в ранговом блоке.

На рис. 2 в виде графика представлены результаты экспериментов. С увеличением количества рабочих процессов в два раза время (t) обработки фрактального алгоритма уменьшается приблизительно вдвое, что подтверждает линейность уменьшения времени сжатия фрактальным методом.

После распределения задач по процессам, каждый процесс выполняет поиск соответствия доменных и ранговых блоков в своей области изображения. В случае деления на меньшие ранговые блоки процессы также самостоятельно сопоставляют меньшие блоки с доменными блоками. Это означает, что в полученной области изображение может быть неоднородно, и для нахождения соответствия ранговый блок может дробиться до минимального размера (2×2 пикселя). В таких случаях возникает неравномерность нагрузки вычислительных процессов, так как, в зависимости от структуры изображения, количество разбиений (а, следовательно, и количество операций) при поиске блоков может отличаться для разных областей изображения

Для примера можно взять изображение frymire. При сжатии изображение на 64 ядрах минимальное время обработки рабочим процессом составляет 129,037 с, а максимальное вычислительное время – 443,582 с.

Тем самым разброс по времени между рабочими процессами составляет 314,545 с. Среднее время продолжительность рабочих процессов составляет 350,251 с, и разница между максимальным и средним временем составляет 93,259 с. При равномерном распределении нагрузки время работы алгоритма уменьшилось бы на 26 %. Это направление требует дальнейших исследований в области методов балансировки нагрузки, например, таких, как указано в [16]. Решение вопросов балансировки вычислительной нагрузки значительно усложняется при изменении количества подзадач (разбиение рангового блока методом квадродерева) в ходе вычислений. Перераспределение подзадач между процессорами уже непосредственно в процессе выполнения параллельной программы может приводить к задержкам (блокировкам), если запрашиваемые данные из очереди сообщений еще не были отправлены процессами-источниками.

В таблице дано сравнение по качеству восстановленного изображения и размеру сжатого файла для параллельной реализации алгоритма фрактального сжатия и для алгоритма JPEG. Сжатие фрактальным методом проводилось максимум на 64-х вычислительных ядрах с организацией 64-х параллельных MPI-процессов.

Как видно из таблицы, на 64-х ядрах можно получить, в среднем, 30-ти кратное ускорение сжатия. Также, следует отметить, что качество восстановленного изображения после фрактального сжатия несколько ниже по критерию PSNR, но выше по критерию SSIM, чем у изображения, сжатого методом JPEG. При этом в отличие от метода, представленного в работе [6], в данной реализации алгоритма не происходит скачков качества для разных изображений, и, в целом, это качество выше (на 20 %). Дисперсия качества восстановленных изображений алгоритма, представленного в работе [6], составляет 12,29 дБ, а в данной реализации алгоритма дисперсия составляет 3,69 дБ. Время сжатия на 64-х ядрах достаточно велико, но, как было указано выше, при увеличении количества вычислительных ядер можно существенно уменьшить время сжатия, так как алгоритм демонстрирует линейное ускорение. Объем сжатого файла меньше, чем у JPEG в среднем на 15 %, при этом следует учесть, что после фрактального сжатия не проводилось кодирование по Хаффману, тогда как в сравниваемой реализации JPEG [17] такое кодирование выходной последовательности проводится.

Выводы

Проведен эксперимент сжатия изображений фрактальным методом на высокопроизводительной вычислительной системе с распределенной памятью. На примере эталонных изображений из стандартной библиотеки показан выигрыш в объеме сжатого файла при сходном качестве по сравнению с алгоритмом JPEG. Подтверждена линейная

зависимость времени сжатия параллельным алгоритмом от числа процессоров (ядер). Выявлен дисбаланс нагрузки вычислительных процессов, вызванный неоднородностью сжимаемых изображений. В ходе дальнейших исследований предполагается рассмотреть методы выравнивания нагрузки вычислительных процессов, а также реализовать и исследовать алгоритм на вычислительных системах с большим количеством процессорных ядер, например, на системах, оснащенных акселераторами GPGPU.

Исследование поддержано грантом по ФЦП «Научные и научно-педагогические кадры инновационной России» (госконтракт № 14.740.11.0398) и проектом 7.701.2011 (НИР 1/12 темплана ТУСУР) по госзаданию Министерства образования и науки.

СПИСОК ЛИТЕРАТУРЫ

1. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2006. – 1072 с.
2. Уэлстид С.Т. Фракталы и вейвлеты для сжатия изображений в действии. – М.: Триумф, 2003. – 320 с.
3. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М.: Диалог-МИФИ, 2003. – 384 с.
4. Денисюк А.А., Полупанов А.А. Фрактальное сжатие изображений // Перспективные информационные технологии и интеллектуальные системы. – 2006. – № 4. – С. 24–29.
5. Илюшин С.В. Ускорение фрактального сжатия изображений путем классификации блоков по полярному углу их центров MACC // T-Comm. Телекоммуникации и транспорт. – 2011. – № 4. – С. 43–47.
6. Шарабайко М.П., Осокин А.Н. Быстродействующий алгоритм фрактального сжатия изображений // Известия Томского политехнического университета. – 2011. – Т. 318. – № 5. – С. 52–57.
7. Barnsley M.F. Fractals Everywhere. – London: Academic Press Inc., 1988. – 534 p.
8. Jacquin A.E. Image coding based on a fractal theory of iterated contractive image transformations // IEEE Trans. on Image Proc. – 1992. – V. 1. – P. 18–30.
9. Fisher Y. Fractal image compression – Theory and Application. – N.Y.: Springer-Verlag, 1994. – 341 p.
10. Гергель В.П., Стронгин Р.Г. Основы параллельных вычислений для многопроцессорных вычислительных систем. 2 изд. – Н. Новгород: ННГУ, 2003. – 184 с.
11. Кулбаев С.С., Бойченко И.В., Голенков В.В. Эффективное сжатие цифровых изображений с применением высокопроизводительных вычислительных систем // VI Сибирская конференция по параллельным и высокопроизводительным вычислениям: Сб. трудов VI Сибирской конференции по параллельным и высокопроизводительным вычислениям. – Томск, 2012. – С. 75–80.
12. Антонов А.С. Параллельное программирование с использованием технологии MPI. – М.: МГУ, 2004. – 74 с.
13. Hore A., Ziou D. Image quality metrics: Psnr vs. ssim // International Conference on Pattern Recognition (ICPR). – Istanbul, Turkey, 2010. – P. 2366–2369.
14. Университетский кластер МСЦ РАН // JSCC RAS Cluster Console. URL: https://unihub.ru/resources/js3console (дата обращения: 19.03.2012).
15. Test image repository // Fractal coding and analysis group. 2011. URL: http://links.uwaterloo.ca/Repository.html (дата обращения: 19.03.2012).
16. Rotaru T., Nageli H.H. Heterogeneous dynamic load balancing with a scheme based on the Laplasian polynomial // Lecture Notes in Computer Sciences. – 2001. – V. 1. – № 1. – P. 107–114.
17. ImageMagick // Convert, edit, and compose images. Studio LLC. 2011. URL: http://imagemagick.org (дата обращения: 15.12.2011).

Поступила 13.07.2012 г.